Real-Time Thevenin Impedance Computation

Stefan Sommer Hjörtur Jóhannsson
Department of Electrical Engineering, Technical University of Denmark
Email: shso@elektro.dtu.dk / hj@elektro.dtu.dk

***Abstract*—Stable and secure operation of power systems becomes increasingly difficult when a large share of the power production is based on distributed and non-controllable renewable energy sources. Real-time stability assessment is dependent on very fast computation of different properties of the grid operating state, and strict time constraints are difficult to adhere to as the complexity of the grid increases. Several suggested approaches for real-time stability assessment require Thevenin impedances to be determined for the observed system conditions. By combining matrix factorization, graph reduction, and parallelization, an algorithm for computing Thevenin impedances an order of magnitude faster than previous approaches is developed. The *factor-and-solve* algorithm is tested with data from several power grids of varying complexity, and experiments show how the algorithm allows real-time stability assessment of complex power grids at millisecond time scale.**

I. INTRODUCTION

Efforts on de-carbonizing the power system often imply a shift from centralized and controllable energy production to distributed and non-controllable renewable energy sources. This shift makes stable and secure operation of power systems an increasingly challenging task.

In traditional power systems, stability could be assessed off-line and sensitivities to various contingencies established by running time-consuming simulations. Multiple factors of the future power system challenge this approach: the complexity of the power grid will rise with increased de-centralization resulting in increased computational burden and longer run-time for simulations; and the power system should be able to operate under rapidly changing conditions for example due to inclusion of weather dependent energy sources. Large fluctuations of the system operating point can be common, and in combination these factors will likely make the results of conventional off-line stability assessment obsolete even before the time-domain simulation has completed. The need for real-time transient stability assessment has therefore been noted by several sources [1].

In this paper, a fast algorithm for real-time computation of Thevenin impedances in complex power systems is developed. Because several approaches for real-time stability assessment require knowledge of Thevenin impedances, the algorithm makes the use of these approaches feasible for complex power systems at increased time resolution. The factor-and-solve algorithm can therefore play a key role in concrete implementations of these transient stability assessment methods. Theoretical arguments for why poor performance is observed for previous approaches is given, and parallelism in parts of the algorithm is exploited to obtain a good balance between serial and parallel computation. The developed method is evaluated on several complex power systems showing how a key property of the power system state can be computed at millisecond scale and used for real-time stability assessment.

A. *Paper Overview*

The paper starts with a short description of approaches to real-time transient stability assessment. In Section III, the problem of computing Thevenin impedances and previous computational approaches are described. LU-factorization of the network admittance matrix and node elimination strategies are discussed in the following section. The factor-and-solve algorithm is developed in Section VI and its performance and validity are evaluated and compared to other approaches in the last section. The paper ends with concluding remarks.

II. BACKGROUND

Transient stability assessment is challenged both by a shift in *time-scale* and by increase in the *complexity* of power systems. With conventional power systems and centralized production, a relatively stable and controllable operating point allowed off-line stability evaluations to remain valid for hours. With de-centralization, weather dependence, and reduced control, stability of the rapidly fluctuating operating point must be evaluated at a much shorter time-scale, preferably in real-time.

At the same time, the complexity of power systems increases. Renewable energy sources add to the number of buses in the system, and power systems becomes increasingly interconnected to even out differences in the production by e.g. weather dependent sources. It is no longer sufficient to consider networks regionally due to increasing number of interconnections across national borders. This in turn requires real-time computation on networks with thousands of buses and tens of thousands of branches.

Real-time monitoring and control is in particular enabled by the advent of phasor measurement units (PMUs, [2], [3]). A few methods have been developed for real-time stability assessment using PMU data. In [4], an existing method is adapted for real-time use while [5], [6] propose a entirely new approach exploiting analytically derived expressions for stability boundaries [7]. See also [8] for a combined off-line/on-line approach. The adaptability of existing off-line

This work is part of the Secure Operation of Sustainable Power Systems (SOSPO) project with support from the Danish Strategic Research Council (DSF).

approaches to real-time computation and the dependence on grid complexity is surveyed in [9].

Thevenin impedance calculations constitutes a major component of the stability assessment in [6]. Several additional methods base voltage stability monitoring on local measurements and Thevenin impedances [10], [11], [12], [13].

Computations involving power grids are often performed on matrices representing the connections and state of the of the network. Typical computations involves factorizing these matrices using solvers that are particularly suited for sparse network computations [14]; relaxation methods [15]; and graph reduction algorithms [16]. See also [17] for comparison of factorization and relaxation methods.

A full time-domain simulation will typically simulate a differential-algebraic system. Thevenin impedance computations differ by only considering the algebraic part of network equations and the computational effort is therefore significantly reduced. This makes Thevenin impedance computations applicable as parts in real-time stability assessment methods. The computational aspects of dealing with this particular system of equations are different than when using the full differential-algebraic system, and the algorithm developed here reflects the real-time requirement.

III. Thevenin Impedances

Consider a power grid consisting of N nodes with the voltage at $M \leq N$ nodes being kept constant by means of voltage control equipment. Letting Y denote the system admittance matrix, the system node voltage equation is

$$I = YV \ . \tag{1}$$

The M voltage controlled nodes (*vc*s) and the $N-M$ nodes of non-controlled voltage (*nc*s) can be ordered so that the *nc*s and *vc*s are numbered by indices $1,\ldots,N-M$ and $N-M+1,\ldots,N$, respectively. The system admittance matrix then takes the form

$$Y = \begin{pmatrix} Y_{nc} & Y_{link} \\ Y_{link}^T & Y_{vc} \end{pmatrix} \tag{2}$$

where Y_{nc} denotes the admittance matrix of only the non-controlled *nc* part of the system, Y_{vc} denotes the admittance matrix of the voltage controlled *vc* part, and Y_{link} encodes the links between the *nc* and *vc* parts of the network.

For each node k of the *vc*s, the aim is to compute the *Thevenin impedance* for the node, i.e. the impedance seen from node k when all *vc* nodes besides node k node are shorted. This situation can be modeled by removing all *vc* nodes besides k from the system, and the Thevenin impedance $Z_{th,k}$ can then be obtained by inverting the resulting admittance matrix. Let $Y_{link,\cdot k}$ denote the column of the link matrix Y_{link} corresponding to the kth node, and, correspondingly, let $Y_{link,k\cdot}^T$ denote the row of the transpose link matrix corresponding to the kth node. Letting $Y_{(k,k)}$ denote the kth diagonal element of Y, define

$$Y_k = \begin{pmatrix} Y_{nc} & Y_{link,\cdot k} \\ Y_{link,k\cdot}^T & Y_{(k,k)} \end{pmatrix} \ ,$$

i.e. the admittance matrix with all *vc* nodes but node k removed. The Thevenin impedance $Z_{th,k}$ then equals the last diagonal element of the inverted matrix Y_k^{-1}.

A naive algorithm for computing $Z_{th,k}$ for all *vc* nodes would set up and invert Y_k for each $k = N-M+1,\ldots,N$. This is a very inefficient approach and in practice infeasible for real-time computation since the number of arithmetic operations required for inverting a matrix has complexity $O(n^3)$ [18][2]. The fact that the upper left $(N-M)\times(N-M)$ submatrix of Y_k does not depend on k strongly suggests more efficient approaches.

IV. Impedances from LU-factorizations

The LU-factorization [18], [19] splits a matrix into a product of a lower diagonal and an upper diagonal matrix, e.g. for the admittance matrices Y and Y_k, factorizations

$$Y = LU \text{ and } Y_k = L_k U_k$$

can be obtained. In particular, the inverse of Y_k is given by $Y_k^{-1} = U_k^{-1} L_k^{-1}$. It is conventional to let the diagonal elements of L_k be all 1 which then implies that the Thevenin impedance $Z_{th,k}$ is given by inverse of the last diagonal element of U_k, i.e. $Z_{th,k} = (U_{k,(N-M+1,N-M+1)})^{-1}$.

It is shown in [5] that this diagonal element and hence $Z_{th,k}$ can be recovered from the factorization L, U of the full admittance matrix Y by the formula

$$U_{k,(N-M+1,N-M+1)} = Y_{(k,k)} - \hat{L}_{k\cdot}\hat{U}_{\cdot k} \tag{3}$$

with the last term being the inner product between the entries $1,\ldots,N-M$ of the kth row of L and of the kth column of U. The advantage of using this relation is that only one matrix, Y, needs to be factorized in order to compute $Z_{th,k}$ for all *vc* nodes, i.e. for all $k = N-M+1,\ldots,N$. Although Y is larger than Y_k, this offers a substantial reduction in computational effort. In [5], LU-factorization of Y and (3) is used for computing Thevenin impedances. This method is analyzed below in order to develop a more efficient approach, and the method is used as basis for the comparisons in the experiment section.

A. Sparsity, Ordering and Fill-Ins

Due to the very high sparsity of network matrices, LU-factorization is in general a very efficient procedure. Though the worst case performance is $O(N^3)$, the complexity is in practice close to linear [17][3]. This complexity can be reached with appropriate ordering of the matrix rows and columns and with specialized solvers.

A key factor in achieving close to linear complexity in the factorization is minimizing the number of *fill-ins*, non-zero elements of the factors L and U that are not present in Y. The number of fill-ins is very dependent on the ordering of the matrix Y. For network matrices, ordering algorithms like Approximated Minimum Degree (AMD, [20]) and variants

[2] Instead of inverting Y_k, a linear system could be solved to get the result. This, however, does not change the $O(n^3)$ complexity.

[3] [17] experimentally assesses the complexity to $O(N^\alpha)$, $\alpha \approx 1.2$.

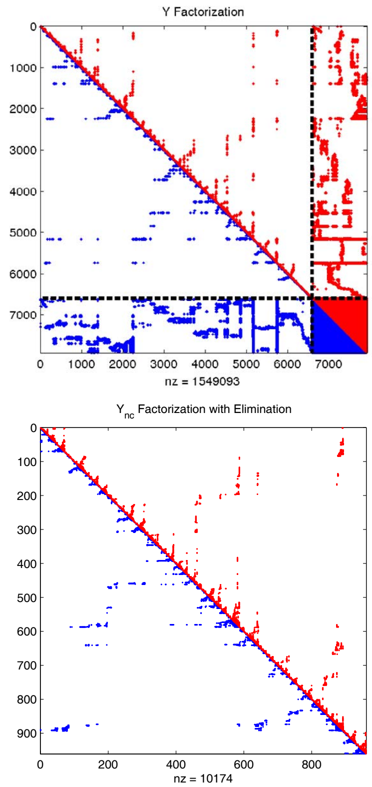

Figure 1. Sparsity patterns of (top) the LU-factorization of the full admittance matrix Y ordered with the voltage controlled nodes (vcs) to the right; and (bottom) the LU-factorization of the Y_{nc} submatrix after application of node elimination. The excessive fill-ins in the lower right vc-part of the full factorization (top) slows down the algorithm. In contrast, the factorization of the reduced matrix (bottom) can be done with very limited fill-in and consequently very fast factorization. The factor-and-solve algorithm reduces both the dimension of the matrix to be factored (in this case from 7917×7917 to 960×960) and the number of non-zeros in the factors (from 1549093 to 10174) thus providing a great reduction in computational time.

ensure a very low degree of fill-in. The number of both non-zeros in Y and additional fill-ins are in practice close to linearly correlated with N which implies the close to linear complexity of the factorization.

In (2), an ordering with the ncs occurring with lower indices than the vcs is used. This ordering is required for the relation (3) that allows the Thevenin impedances to be extracted. To adhere to this indexing convention, [5] applies AMD ordering to the submatrices Y_{nc} and Y_{vc} individually before combining them to obtain the full matrix Y as in (2). The result of this partial ordering strategy is that the upper left part of the factors L, U becomes adequately sparse but the lower right part of the factors unfortunately contains a very large number of fill-ins. This problem that slows down the algorithm considerably is illustrated in Figure 1. In the next section, a theoretical explanation for the occurrence of the fill-ins is given, and it is shown how an improved Thevenin impedance algorithm avoids this problem.

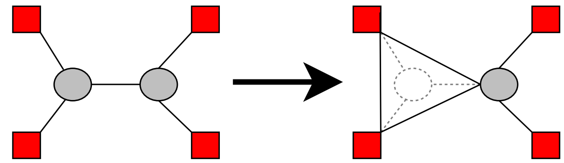

Figure 2. Elimination of one of two interior nodes in a six node network. The node to be eliminated has degree three, and with three new branches added to the reduced network, the total number of branches is kept constant. This preserves the sparsity. Elimination of nodes with higher degree will result in an increased number of branches thus reducing sparsity.

V. Schur Complement and Node Elimination

With system loads represented by their admittance values, no current enters the nodes of non-controlled voltage (ncs), and the network equation (1) can be stated as

$$\begin{pmatrix} 0 \\ I_{vc} \end{pmatrix} = \begin{pmatrix} Y_{nc} & Y_{link} \\ Y_{link}^T & Y_{vc} \end{pmatrix} \begin{pmatrix} V_{nc} \\ V_{vc} \end{pmatrix} . \tag{4}$$

Using the *Schur complement* [21], [16]

$$S = Y_{vc} - Y_{link}^T Y_{nc}^{-1} Y_{link}$$

of Y_{vc}, the vc-part of the solution to (4) can be obtained by solving the reduced system $I_{vc} = S V_{vc}$. The Schur complement can in addition be obtained by successively *eliminating* nodes from the system and creating reduced admittance matrices. For each node to be eliminated as illustrated in Figure 2, the new admittance matrix is given by the formula

$$Y_{(i,j)}^{\text{new}} = Y_{(i,j)} - \frac{Y_{(i,N)} Y_{(N,j)}}{Y_{(N,N)}} , \tag{5}$$

and S is the matrix resulting from eliminating all ncs. Confer [16] for more information on node elimination and network reduction.

A. Why Y Should Not Be Factored

When eliminating nodes, branches are added to the resulting network, and in the completely reduced network consisting of all vcs, all pairs of nodes are in general connected by branches. The Schur complement S is therefore a dense matrix.

In [22], [23], it is observed that if a matrix with the block structure (2) is LU-factored, the product of the lower right blocks L_{vc}, U_{vc} of the factors L, U corresponding to the vcs provide the Schur complement of Y_{vc} directly, i.e. $S = L_{vc} U_{vc}$. This provides a way to compute and factor S but it also explains why the large number of fill-ins are observed when computing Thevenin impedances with the method of [5] that uses factorization of the full matrix Y: because S is dense, the factors L_{vc} and U_{vc} will in general not be sparse[4], and L_{vc}, U_{vc} are precisely the lower right blocks of the factors L, U where the excessive number of fill-ins occur. Indeed, any fixed bound on the maximum node degree in both L_{vc} and U_{vc} would imply that the number of non-zeros in S would grow linearly with

[4]The product of sparse matrices can be dense. However, if the node degree of the networks represented by the factors is limited, the product will be sparse.

the number of *vc*s, i.e. M. Since S is dense, the number of non-zeros grow quadratically, $\mathrm{nnz}(S) \approx M^2$, implying that no such bound can exist.

VI. Factor-and-solve Thevenin Impedance Algorithm

Here, a fast algorithm for computing Thevenin impedances is derived that avoids factorization of the full admittance matrix Y and that thereby avoids the excessive fill-in in the Schur complement part of the factors. The resulting algorithm is denoted *factor-and-solve* relating to its composition of two individual steps.

The algorithm is derived by coupling a variant of the relation (3) with the structure of left-looking LU-factorization algorithms. First, a close variant of (3) for computing $U_{k,(N-M+1,N-M+1)}$ uses Y_k instead of Y. Using the factorization $Y_k = L_k U_k$,

$$U_{k,(N-M+1,N-M+1)} = Y_{(k,k)} - \hat{L}_{k,(N-M+1)\cdot} \hat{U}_{k,\cdot(N-M+1)} \tag{6}$$

where the notation in the rightmost term denotes the inner product between entries $1, \ldots, N-M$ of the last row of L_k and of the rightmost column of U_k. The advantage of using this formula is that $\hat{L}_{k,(N-M+1)\cdot}$ and $\hat{U}_{k,\cdot(N-M+1)}$ can be obtained from a factorization $Y_{nc} = L_{nc} U_{nc}$ of the *nc*-part of Y only.

Iterations of left-looking LU-factorization algorithms [19] are now considered. With this class of algorithms, the $N - M + 1$ columns in a factorization $Y_k = L_k U_k$ are computed iteratively from left to right, i.e. starting with column 1 and ending with column $N - M + 1$. At each step j, the upper left $(j-1)$-block of L_k is used to compute the first $j - 1$ entries of the jth column of U_k. In particular, computation of $N - M$ entries of the rightmost column uses only the upper left $(N - M)$-block of L_k, i.e. the block representing the *nc*s. Writing this last step of the algorithm explicitly, the $N - M$ first entries of column $N - M + 1$ of U_k satisfies

$$L_{nc} \hat{U}_{k,\cdot(N-M+1)} = \hat{Y}_{link,\cdot k} \tag{7}$$

where $\hat{Y}_{link,\cdot k}$ denotes the first $N - M$ entries of the column $Y_{link,\cdot,k}$. The vector $\hat{U}_{k,\cdot(N-M+1)}$ is therefore computed with a triangular forward solve using the factorization of Y_{nc} only. Similarly, the first $N - M$ entries of row $N - M + 1$ of L_k can be obtained by the equation

$$U_{nc}^T \hat{L}_{k,(N-M+1)\cdot}^T = \hat{Y}_{link,k\cdot}^T \tag{8}$$

again using only the factorization of Y_{nc}. Thus, using (6), $U_{k,(N-M+1,N-M+1)}$ can be obtained from two forward solutions using the factorization of Y_{nc}.

With the above computation, all matrices and operations involved are sparse and the fill-in producing factorization of the full admittance matrix Y is completely avoided. In addition, the triangular matrices used for the forward solves are not dependent on k, and the factorization of Y_{nc} must therefore be done only once. Due to the sparsity, the forwards solves are each computationally lightweight, and they can in addition be computed completely in parallel. In the sequel, the factorization of Y_{nc} is denoted the *factorization step* and the forward solutions (7),(8) the *forward solve step*. The algorithm for computing Thevenin impedances with this approach is listed in Algorithm 1. Though the experiments section will show that the forward solve step can dominate the runtime, the completely parallel nature of the loop over all *vc*s makes speeding up this step straight-forward by splitting the computation of several compute cores. In contrast, the factorization step is hard to parallelize and therefore in reality the limiting factor of the algorithm. This step is analyzed below.

Algorithm 1 Factor-and-solve Thevenin impedance algorithm.

```
L_nc, U_nc ← factorization of Y_nc
for k = N − M + 1 → N do        ▷ for each vc possibly in parallel
    Û_{k,·(N−M+1)} ← solve(L_nc, Ŷ_{link,·k})
    L̂^T_{k,(N−M+1)·} ← solve(U^T_nc, Ŷ^T_{link,k·})
    U_{k,(N−M+1,N−M+1)} ←
            Y_{k,(k,k)} − L̂_{k,(N−M+1)·} Û_{k,·(N−M+1)}
    Z_{th,k} ← U^{−1}_{k,(N−M+1,N−M+1)}        ▷ Thevenin impedance node k
end for
```

A. Node Elimination and Factorization Speed

The factorization step of Algorithm 1 consist of the LU-factorization of Y_{nc}. The KLU solver [14] is used for the factorization in contrast to e.g. [5] which uses UMFPACK [24] when factoring Y. The sparsity of the network matrices are so high that KLU being a left-looking solver performs better than a right-looking multifrontal methods such as UMFPACK.

KLU is a state-of-the-art and very optimized solver, and it is therefore inherently difficult to improve the factorization speed. Nevertheless, it turns out that the execution time of the factorization step of Algorithm 1 can be reduced by using that only the solution to (6) is needed for which factorization of the full submatrix Y_{nc} is not required. Instead, node elimination prior to factorization is performed in order to reduce the matrix size. This part of the factor-and-solve algorithm is denoted the *node elimination* step.

Successive node elimination using the update formula (5) produces an equivalent network matrix that has fewer nodes but potentially is less sparse. For simulation of large resistor networks, several methods elimination parts of the system is used to reduce the size of the network as much as possible without producing to much fill-in [25], [26].

For the applications considered here, node elimination can speed up the computation but only if careful consideration is taken with respect to the amount of fill-ins and the time used for elimination. Due to the efficiency of KLU, the algorithm can be quite relaxed in removing only a relatively limited number of nodes. This is done with a simple fill-in reducing strategy: the algorithm scans through the *nc*s removing a node only if it is connected to less than 4 other *nc*s and if the fill introduced in the link matrix Y_{link} is limited. Since removing nodes of degree 3 or less does not introduce fill-ins, this strategy ensures that the number of non-zeros in Y_{nc} does not increase during the process, confer Figure 2. The number of non-zeros in Y_{link} will in general increase but the number of added fill-ins is controlled by a fixed limit.

It will be shown in the experiments section that the application of node elimination prior to running Algorithm 1 reduces the computational effort for the factorization step by a factor of 2-3.

VII. Experiments

In this section, the speedup provided by the factor-and-solve Thevenin impedance algorithm, its absolute runtime, and its validity is evaluated. In particular, the experiments will show that the Thevenin impedance of all generators for power systems of considerable sizes can be established in less than 3 ms. In addition, the runtime of the serial and parallel parts of the algorithm will be explored in order to evaluate the achieved overall efficiency, and a great reduction in size and number of non-zeroes for the matrix to be factored will be observed. The factor-and-solve algorithm will be compared to the method of [5], that, to the best of our knowledge, is the only computational method for computing Thevenin impedances described in the literature. The validity of the algorithm is ensured by measuring the differences between the methods. For all experiments, it is observed that the results are equal up to numerical precision showing that the new algorithms produces correct results.

The experiments are performed on admittance matrices generated from test systems included in the PSS®E-30.0[5] and MATPOWER [27] network simulation packages. The test systems include the US west-coast (1648 buses, 2602 branches) and US east-cost (7917 buses, 13014 branches) power grids along with 6 additional systems ranging from 2383 to 3120 buses[6].

The runtime is tested on a 3.2GHz Intel Core i7 hexa-core desktop CPU. In accordance with [5], UMFPACK [14] is used for factoring the full admittance matrix with the reference method, and KLU [14] is used for the factorization step of Algorithm 1. The main loop of the algorithm is parallelized using six threads, and the node elimination step uses AVX vector instructions to exploit fine-grained parallelism.

Figure 3 shows for each test system the runtime of the original Thevenin impedance algorithm employing LU-factorization of the full admittance matrix, the runtime of the factor-and-solve algorithm without node elimination, and the factor-and-solve algorithm with node elimination prior to the factorization. For all three approaches, the runtime of the initial symbolic pre-factorization step is left out of the measurements because this step only needs to be done once for each network. The timings are performed just on the computational parts leaving out the time used for initial copying of data, and the obtained timings are averaged over a large number of runs. Please note the logarithmic scale on the vertical axis and the achieved approximately 80 times speedup on the largest system with the factor-and-solve algorithm compared to the previous method.

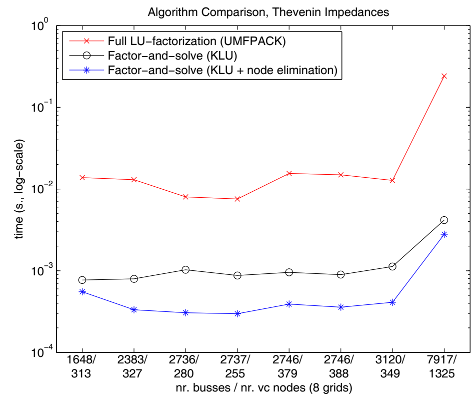

Figure 3. Computation time for determining Thevenin impedances using the full LU-factorization ([5], red), the factor-and-solve algorithm (black), and the factor-and-solve algorithm with node elimination (blue). Evaluation performed on 8 power grids ranging from 1648 buses to 7917 buses with between 313 and 1325 voltage controlled nodes. Note the logarithmic scale on the time axis. For the largest system, the new method is roughly 80 times faster than the previous approach.

In Figure 4, the runtime of the three different parts of the factor-and-solve algorithm is plotted: node elimination, factorization, and forward solve. It is seen that a relatively large portion of the computational effort is spend on the forward solve. It is important to relate this to the fact that the forward solve step can be run in parallel. For the results here, all 6 cores of the test machine are used. If a reduction in runtime is needed, a machine with more cores will allow the runtime of the forward solve step to be reduced to less than the runtime used for the reduction and factorization. In addition, there is room for more optimization of the code used for computing the forward solutions.

Because the forward solve step can be parallelized, the serial parts of the algorithm are in reality the true bottlenecks. In Figure 5, the runtime of the serial parts are plotted in order to evaluate the benefits of the node elimination step. Employing node elimination results in a 2-3 times speedup for this part of the algorithm. In total, the factor-and-solve algorithm reduced the dimension of the matrix to be factorized for the largest test system from 7917×7917 (the full admittance matrix) to 960×960 (the non-controlled part of the admittance matrix after node elimination). At the same time, the number of non-zeros in the factors is reduced from 1549093 to 10174.

VIII. Conclusion

Real-time calculation of Thevenin impedances is important for several suggested approaches to stability assessment. In the paper, theoretical arguments for why excessive fill-in occurs

[5]http://www.energy.siemens.com/us/en/services/power-transmission-distribution/power-technologies-international/software-solutions/pss-e.htm

[6]The packages includes test systems that are considerably smaller. The runtime for these systems are negligible with the developed algorithm and therefore not included in the evaluation.

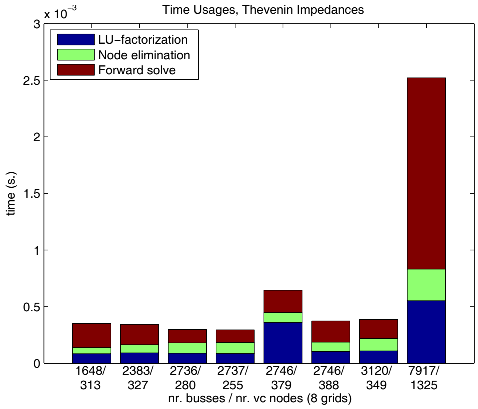

Figure 4. The time consumed for the three different parts of the factor-and-solve algorithm: factorization (blue), node elimination (green), and forward solve (red). The forward solve step parallelizes completely and the runtime can thus be reduced by employing more computational cores.

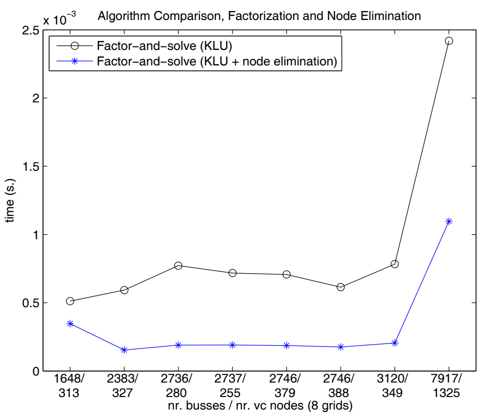

Figure 5. Runtime for the factor-and-solve algorithm excluding the forward solve step, without node-elimination (black) and with node-elimination (blue). Node elimination results in a speedup of a factor 2-3 for this part of the algorithm.

in the factorization used for previous approaches to calculating Thevenin impedances are given. These insights lead to an improved algorithm that achieves an order of magnitude speedup and that allows parallelization of the computationally heavy part of the algorithm. Additional saving in computation time is in achieved by using node elimination prior to the factorization step.

The performance of the factor-and-solve algorithm is evaluate on admittance matrices representing large and complex power grids. Comparison with previous approaches shows approximately 80 times speedup for the largest power system. In addition, it is determined how the different steps of the algorithm affect its performance and how the runtime can be controlled using parallelization of the forward solve step. As a result, for these systems, the Thevenin impedance computation is no longer a bottleneck for real-time transient stability assessment.

References

[1] F. Li, W. Qiao, H. Sun, H. Wan, J. Wang, Y. Xia, Z. Xu, and P. Zhang, "Smart transmission grid: Vision and framework," *IEEE Transactions on Smart Grid*, vol. 1, no. 2, Sep. 2010.

[2] A. G. Phadke and J. S. Thorp, *Synchronized Phasor Measurements and Their Applications*, 1st ed. Springer, Sep. 2008.

[3] A. Phadke and R. de Moraes, "The wide world of wide-area measurement," *IEEE Power and Energy Magazine*, vol. 6, no. 5, Oct. 2008.

[4] M. Glavic and T. Van Cutsem, "Wide-area detection of voltage instability from synchronized phasor measurements. part i: Principle," *Power Systems, IEEE Transactions on*, vol. 24, no. 3, Aug. 2009.

[5] H. Jóhannsson, "Development of early warning methods for electric power systems," Ph.D. dissertation, Technical Univ. of Denmark, 2011.

[6] H. Johannsson, R. Garcia-Valle, J. Weckesser, A. Nielsen, and J. Østergaard, "Real-time stability assessment based on synchrophasors," in *PowerTech, 2011 IEEE Trondheim*, Jun. 2011.

[7] H. Jóhannsson, J. Østergaard, and A. H. Nielsen, "Identification of critical transmission limits in injection impedance plane," *International Journal of Electrical Power & Energy Systems*, vol. 43, no. 1, 2012.

[8] Y. V. Makarov, P. Du, S. Lu, T. B. Nguyen, J. Burns, and J. Gronquist, "Wide-area dynamic security region," in *NAPS, 2009*, Oct. 2009.

[9] T. Weckesser, H. Jóhannsson, S. Sommer, and J. Østergaard, "Investigation of the adaptability of transient stability assessment methods to real-time operation," in *IEEE PES ISGT Europe*, Berlin, 2012.

[10] S. Corsi and G. Taranto, "A real-time voltage instability identification algorithm based on local phasor measurements," *IEEE Transactions on Power Systems*, vol. 23, no. 3, Aug. 2008.

[11] L. Warland and A. Holen, "Estimation of distance to voltage collapse: Testing an algorithm based on local measurements," in *PSCC*, Sevilla, 2002.

[12] I. Smon, G. Verbic, and F. Gubina, "Local voltage-stability index using tellegen's theorem," in *IEEE Power Engineering Society General Meeting, 2007*, Jun. 2007.

[13] K. Vu, M. Begovic, D. Novosel, and M. Saha, "Use of local measurements to estimate voltage-stability margin," *IEEE Transactions on Power Systems*, vol. 14, no. 3, Aug. 1999.

[14] T. A. Davis and E. Palamadai Natarajan, "Algorithm 907: KLU, a direct sparse solver for circuit simulation problems," *ACM Trans. Math. Softw.*, vol. 37, no. 3, Sep. 2010.

[15] M. Ilic'-Spong, M. L. Crow, and M. A. Pai, "Transient stability simulation by waveform relaxation methods," *Power Systems, IEEE Transactions on*, vol. 2, no. 4, Nov. 1987.

[16] F. Dorfler and F. Bullo, "Kron reduction of graphs with applications to electrical networks," *IEEE Transactions on Circuits and Systems*, 2011.

[17] F. Pruvost, T. Cadeau, P. Laurent-Gengoux, F. Magoules, and F.-X. Bouchez, "Numerical accelerations for power systems transient stability simulations," in *17th Power System Computation Conference*, 2011.

[18] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction To Algorithms*. MIT Press, 2001.

[19] T. A. Davis, *Direct Methods for Sparse Linear Systems*. SIAM, Sep. 2006.

[20] P. R. Amestoy, T. A. Davis, and I. S. Duff, "Algorithm 837: AMD, an approximate minimum degree ordering algorithm," *ACM Trans. Math. Softw.*, vol. 30, no. 3, Sep. 2004.

[21] F. Zhang, Ed., *The Schur Complement and Its Applications*, ser. Numerical Methods and Algorithms, 2005, vol. 4.

[22] Y. Saad, *Iterative Methods for Sparse Linear Systems*. SIAM, Apr. 2003.

[23] Y. Saad and M. Sosonkina, "Distributed schur complement techniques for general sparse linear systems," *SIAM J. SCI. COMPUT*, vol. 21, 1997.

[24] T. A. Davis, "Algorithm 832: UMFPACK v4.3—an unsymmetric-pattern multifrontal method," *ACM Trans. Math. Softw.*, vol. 30, no. 2, Jun. 2004.

[25] J. Rommes and W. H. A. Schilders, "Efficient methods for large resistor networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 1, Jan. 2010.

[26] Z. Ye, D. Vasilyev, Z. Zhu, and J. Phillips, "Sparse implicit projection (SIP) for reduction of general many-terminal networks," in *IEEE/ACM International Conference on Computer-Aided Design*, Nov. 2008.

[27] R. Zimmerman, C. Murillo-Sanchez, and R. Thomas, "MATPOWER: steady-state operations, planning, and analysis tools for power systems research and education," *IEEE Transactions on Power Systems*, vol. 26, no. 1, Feb. 2011.